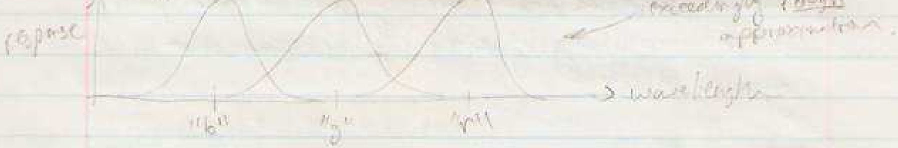
# Colors

## How Colors are Perceived

The eye has 3 types of cone cells, each has a sensitivity peak at a different wavelength of light, usually referred to as red, green, & blue receptors although the peaks don't really correspond to these colors.




So when light hits a red object, the red wavelengths get reflected (everything else is absorbed, ideally) & strike your eye, triggering a response big in the "red" cones & little or no response in the other cones.

The receptors' sensitivity ranges overlap so when some other color comes in, it will trigger a response in both receptors, & that combined response is what tells your brain you're seeing that color.

## RGB Color model

It's pretty hard to create pixels that can generate any wavelength of light on demand. So to generate different colors, we use different ratios of red, green, & blue light to stimulate the three different types of cones at ratios that match the ratios excited by the intended wavelength.

→

Depending on the exact primary colors (R, G, & B), it may not be possible to generate the full space of colors perceptible to the human eye w/ RGB color.

## Other Color Encodings (HSV, HSL, $YC_bC_r$)

These different color palettes and selectors you see on computer programs are ~~+~~ actually different color spaces, just different ways of encoding the same color space (typically RGB).

HSV + HSL are intended to be more intuitive than RGB. H is hue, which specifies the color, red, green, blue or somewhere in between. S is saturation, which is basically how vibrant the selected hue is; 0 saturation is all gray scale. Then V is value + L is lightness, which are two different ways of encoding essentially how light or dark the color is. With HSV + HSL, it's easier to adjust human-relevant parameters like saturation w/out changing hue + lightness, etc.

$YC_bC_r$ is even less intuitive than RGB, but is designed to be more efficient than other encodings. Our vision is more sensitive to the luma (grayscale) aspects of an image than to the color, but the luma of an RGB image is a function of the R, G, + B channels. So YCC stores the Luma, Red, + Blue data (but indirectly), + then calculates the green from that.

→

That way, we can use more bits to store the important luma channel & fewer to store the less important color channels. Further it doesn't store R + B directly, it stores the difference between the channels & the luma channel. Often these differences will be in a fairly narrow range compared to the actual raw color channels, allowing more opportunity for compression.